DOCUMENTAÇÂO – TP ALLEGRO

*Aluna:* Letícia da Silva Macedo Alves

*Disciplina:* Algoritmos e Estruturas de Dados I

*Professor*: Pedro O.S. Vaz de Melo

* **Introdução**

A proposta do Trabalho Prático foi desenvolver um jogo semelhante ao conhecido “Enduro” utilizando a linguagem C e a biblioteca Allegro. Assim, podendo consolidar os conhecimentos e habilidades desenvolvidas durante o curso de AEDS1. No jogo temos um carro principal que pode ser controlado pelo jogador com as teclas A (movendo-o para a esquerda) e D (para a direita). Eventualmente carros “inimigos” aparecem na outra extremidade da pista e caminham na direção contrária a do jogador, que deve, então, desviar. A cada desvio acumula-se pontos e em caso de colisão o jogo é finalizado e tem-se o score obtido na partida.

* **Funções e procedimentos:**

***int sorteiacor():*** gerar números aleatórios entre 0 e 255, atribuídos na criação dos carros (base no sistema de cores RGB da biblioteca);

***float numfloat (int min, int max):*** gerar número float aleatório entre um valor mínimo e um máximo (passados como parâmetro);

***void criaCarro(Car \*c, float x):*** recebe como parâmetros um carro por referência e a coordenada x de seu centro e desenha o carro nas novas coordenadas (com base na movimentação) a cada instante de tempo;

***float getRatio (float y):*** responsável por calcular uma razão de proporcionalidade que define a taxa para o ajuste das coordenadas x a cada instante para dar o efeito dos carros inimigos se aproximando pela pista em relação ao jogador. Essa taxa multiplica todas as dimensões do carro (largura e altura da base e das rodas);

***float modulo(float n):*** devolve o valor absoluto de um número e foi utilizada na função que checa se houve colisão entre os carros inimigos e o do jogador;

***int colisao (Car \*inim, float x):*** responsável por checar a cada instante se houve uma colisão entre o carro do jogador e os demais. Em caso afirmativo retornará 1 (e o jogo será finalizado), caso contrário, 0;

***void init\_global\_vars():*** são atribuídos valores a algumas variáveis com base em cálculos envolvendo as proporções e medidas de tela e cenário. Tais variáveis podem ser utilizadas durante todo o código. Além disso, aqui os valores iniciais dos campos de cada elemento (estruturas Car) do vetor são preenchidos;

***void draw\_scenario(ALLEGRO\_DISPLAY \*display):*** desenha o cenário inicial e o redesenha a cada ciclo do temporizador (com base nos novos parâmetros atualizados dos carros);

***int main(int argc, char \*\*argv):*** contém a maior parte das “especificações” da biblioteca alegro como rotinas de inicialização e criação da fila de eventos. Nele está a parcela do código responsável por checar se as teclas A e D estão sendo pressionadas (movimentação do carro do jogador). Também contém a verificação do temporizador. Assim, se o tempo passa, uma série de ações e chamadas são executadas (como draw\_scenario, atualizando a tela de jogo);

* **Implementações e funcionamento**

1. **Carro do jogador**

Desenhado com 3 retângulos e tem o centro do desenho (retângulo do meio) como coordenada “base” para seu funcionamento. Seu movimento é controlado com as teclas A e D, que o movem, respectivamente, para a esquerda e para a direita. Para isso temos as variáveis:

float x\_centro, y\_centro;

int moveesq, movedir;

Em “int main(int argc, char \*\*argv)”, temos a parte do código (composta por comandos “if” e “switch” responsável por checar se alguma dessas teclas está ou não sendo pressionada. Se A estiver sendo pressionada, por exemplo, “moveesq” receberá o valor 1 (enquanto movedir permanecerá com 0) e o valor de x (do centro) será decrementado de modo que o carro será a cada instante redesenhado mais para a esquerda.

1. **Carros inimigos**

Os carros são desenhados com 3 retângulos, sendo os das laterais as rodas. Para a criação dos carros inimigos foi usada uma estrutura de dados do tipo Car composta com os seguntes campos:

float x, y, acel; //Coordenadas (x,y) do centro do desenho e aceleração  
  
int R, G, B; //Representando a composição (aleatória) da cor do carro

float tam, tamr, alt, altr; //Largura e altura (dimensões) do carro e suas rodas

São então armazenados em um vetor (do tipo Car) de NUMCARS elementos (no caso 10) e os campos preenchidos inicialmente em “void init\_global\_vars()”.

1. **Crescimento dos inimigos**

Para dar a impressão de que os carros adversários estão caminhando na direção contrária a do principal e se aproximando do jogador, temos as variáveis:

float dist, xc, taxa;

float THETA;

e a função “ float getRatio (float y)”, chamada no “int main” passando como parâmetro “dist”. Essa função recebe, a cada instante, a distância (y) entre o carro inimigo e a altura do céu (SKY\_H). Como temos THETA (que é o ângulo que a base da pista faz com as linhas de sua borda), calculamos a taxa (retornada pela função, com base em uma semelhança de triângulos e que é um valor entre 0 e 1) de proporção para as novas dimensões e coordenadas do carro a cada incremento de sua ordenada y. As proporções do carro (larguras e alturas) serão sempre multiplicadas por essa taxa. Para que a proporção entre a distância do carro e da borda da pista seja mantida durante o percurso, usa-se como referência a coordenada x da metade da largura da tela. Assim, o novo x do carro:

xc = (SCREEN\_W/2)+((inim[i].x-(SCREEN\_W/2))\*taxa);

1. **Contagem do score**

A cada carro adversário que ultrapassa os limites de altura da tela, para os quais não houve colisão, a variável score é incrementada em 1.

1. **Recorde**

Ao final de cada jogo, compara-se o score obtido com o maior score obtido  
dentre todas as partidas executadas. Esse recorde está armazenado em um arquivo. Assim, no programa, ao ler este arquivo, caso o score obtido seja maior do que o recorde, o score é escrito no arquivo, substituindo o antigo valor do recorde.

1. **Colisão**

Para identificar uma colisão, utilizamos a função “int colisao (Car \*inim, float x)”, que recebe como parâmetros um carro adversário por referência e o x central do carro do jogador. A colisão deve ser analisada por dois aspectos: colisão superior e colição lateral. Assim, quando a diferença entre os y do centro dos dois carros analisados é menor do que a altura da roda do carro do jogador, se a diferença entre as coordenadas x do centro é menor do que a largura do carro principal, houve colisão. Essa verificação ocorre no temporizador e quando é constatada col=1 e playing=0, finalizando o jogo e mostrando em uma tela preta o score total do jogador e o recorde.

* **Considerações finais (pessoais)**

O trabalho foi desafiador e enriquecedor. Antes do curso não tive experiência alguma com programação. Logo, a percepção própria de crescimento é notória. Sinto que pude desenvolver relações entre as matérias que tive durante o período e relacionar equações matemáticas e raciocínio lógico intrínseco à programação.